Amendments to the Specification

Please replace the paragraph on Page 1, lines 4 - 9 with the following marked-up replacement paragraph:

-- This application is related to the applications having serial numbers 09/422,492 entitled ~~numbers 09/_____ entitled~~ "Selective Data Encryption Using Style Sheet Processing for Decryption by a Client ~~Proxy", 09/_____ entitled~~ Proxy", 09/422,537 entitled "Selective Data Encryption Using Style Sheet Processing for Decryption by a Group Clerk", ~~and 09/_____ entitled~~ and 09/422,431 entitled "Selective Data Encryption Using Style Sheet Processing for Decryption by a Key Recovery Agent", all assigned to the same assignee and filed concurrently herewith on October 21, 1999. --

Please replace the paragraph that begins on Page 9, line 15 and carries over to Page 10, line 7 with the following marked-up replacement paragraph:

-- Commonly-assigned U. S. Patent _____ (serial number 09/240,387, filed 01/29/1999), titled "Method, System, and Apparatus for Selecting Encryption Levels Based on Policy Profiling" suggests tagging data elements in Extensible Markup Language ("XML") documents with field-level or record-level security information. ~~("XML" is a trademark of Massachusetts Institute of Technology.)~~ By inspecting this security-level information and consulting directory entries concerning an individual's access privileges, a server responding to a document request suppresses any document elements for which the requester is unauthorized, determines the encryption algorithm and key length required by the most restrictive remaining element (i.e. the remaining element having the highest-level security requirements), and encrypts

the entire resulting filtered document accordingly. This invention does not solve the problem of encrypted documents with multiple authorized receivers and agents, each with a different need-to-know (i.e. it does not restrict the ability to read certain fields of a document to certain individuals or groups). Nor does it address the problem of client devices with insufficient processing power to decrypt received documents. --

Please replace the paragraph on Page 10, lines 8 - 19 with the following marked-up replacement paragraph:

-- Several solutions for distributing encrypted key material along with the encrypted document to which the key applies are known in the art. The SMIME industry standard defined by the IETF is used in secure e-mail transmission, providing an encapsulation of digitally signed and encrypted objects. (See SMIME charter information that is available from the IETF ~~http://www.ietf.org/html.charters/smime-charter.html on the Web~~ for more information.) The Lotus Notes® software uses a proprietary implementation for key distribution. (~~See http://www.lotus.com on the Web for more information about Lotus Notes.~~ "Lotus Notes" is a registered trademark of Lotus Development Corporation and/or IBM, and more information about Lotus Notes is available by contacting IBM.) However, neither of these existing approaches suggests that individual document fields be encrypted (and other fields not encrypted). Nor do they suggest having different authorized viewing communities, or using multiple and/or different encryption algorithms and/or ~~keys for~~ keys, for different fields in a document that need different levels of security (nor is a capability for distributing multiple keys per document available). --

Please replace the paragraph on Page 25, lines 15 - 21 with the following marked-up replacement paragraph:

-- Commonly-assigned U. S. ~~Patent _____ (serial~~ Patent 6,585,778 (serial number 09/385,899, filed 08/30/1999), titled "Enforcing Data Policy Using Style Sheet Processing", discloses a technique for controlling the content of a document using stored policy information. This invention, referred to hereinafter as the "referenced invention", is incorporated herein by reference. The present invention defines an extension to the stored policy objects defined in this referenced invention, whereby the stored policy objects further comprise attributes specifying the element visibility information described above. These extensions will be described in more detail with reference to Figs. 7A through 7C. --

Please replace the paragraph on Page 26, lines 1 - 17 with the following marked-up replacement paragraph:

-- The employee record example previously discussed will be used to illustrate the benefits as well as the implementation of the present invention. Suppose a company maintains a database (or other repository) of information about its employees, and further suppose that the stored record for each employee comprises the employee's name, employee serial number, data of hire, current salary, and any pertinent medical conditions. Fig. 3 depicts an example of a DTD 300 that may be used to describe the data in the record for an employee of this company. As is well known in the art, a DTD is a definition of the structure of an XML document, and is encoded in a file which is intended to be processed, along with the file containing a particular XML

document, by an XML parser. The DTD tells the parser how to interpret the document which was created according to that DTD. (Note that while the present invention is described with reference to information specified in a DTD, the information may be specified in other semantically equivalent forms. In particular, the schemas which are currently being standardized by the World Wide Web Consortium or "W3C" may be used instead of DTDs. Refer to "XML Schema Part 1: Structures", which is available from the W3C, for ~~can be found on the Internet at http://www.w3.org/TR/xmlschema-1/, for~~ more information. References herein to a DTD are to be interpreted as applying equally to a schema.) This DTD 300 includes entries for the employee name ("empl_name") 350, the employee serial number ("ser_nbr") 360, "date_of_hire" 370, current salary ("curr_salary") 380, and ~~"curr_salary" 380, and~~ "medical_condition" 390.

Please replace the paragraph on Page 28, lines 8 - 19 with the following marked-up replacement paragraph:

-- Another policy used with the employee record example is to limit access ~~to employee's~~ to an employee's current salary to the employee himself, any managers of the company, and any employees within the company's human resources (HR) department. The URI for this policy has been given the entity name "empl_mgr_hr" 311, and is specified 385 in the attribute list declaration for curr_salary 380. The stored policy object located at URI 312 will specify the encryption strength deemed to be appropriate for protecting this employee salary information from unauthorized access. The community attribute in the policy object will preferably comprise three distinguished name values - one for the individual employee, one for the group comprising all managers, and one for the group comprising all employees in the HR department.

(Alternatively, a separate DN entry could be specified for each member of the managers group and/or each member of the HR department, but as previously stated, it is preferable to represent all members of a group by a group DN when the group DN is available.) --

Please replace the paragraph that begins on Page 31, line 6 and carries over to Page 32, line 10 with the following marked-up replacement paragraph:

-- Skipping for now the discussion of Fig. 4B (comprising Figs. 4B1 and 4B2) and proceeding to Fig. 4C (comprising Figs. 4C1 and 4C2), a representative example of a selectively-encrypted document containing the information from source document 400 is shown at 450. Using the policy and element visibility examples discussed with reference to Fig. 3, if the requesting user is the employee 402, this user will be able to recover (i.e. decrypt) the protected values of both curr_salary 408 and medical_condition 410 from the encrypted fields 452, 454 (where these fields 452, 454 contain values used for illustrative purposes only) of document 450 which is transmitted in response to his request. In addition, because selectively-encrypted documents are not customized for a particular requesting user according to the present invention, but rather carry sufficient key distribution material to enable access by any authorized user, employee 402 will be able to recover the protected values 453, 455 from fields 452, 454 even if he was not the original requester of the encrypted document. Similarly, a user who is a manager in this company or who works in the HR department will be able to recover the value 453 of encrypted field 452 if he receives document 450, because these users are within the authorized community for corresponding document element 408, and a user in the medical department will be able to recover the value 455 of encrypted field 454. If a user who is not a manager, is not the

individual employee, and does not work in either the HR or medical department receives encrypted document 450, this user will only be able to view the values of unrestricted document elements 402, 404, and 406, even though the values of the curr_salary and medical_condition elements are contained within this user's copy of the document 450. The manner in which an augmented style sheet processor applies the data policy and visibility rules to yield these results according to the present invention will be discussed below. (Style sheet processing may perform additional changes to source document 400 in the process of generating an encrypted document, such as formatting the employee record information into a predetermined layout or performing target-specific transformations unrelated to data policy and element visibility, using techniques which are known in the ~~art, and~~ art and do not form part of the present invention.) --

Please replace the paragraph on Page 35, lines 11 - 13 with the following marked-up replacement paragraph:

-- Key class objects 530 corresponding to each preprocessing key class object 520 are built during the post processing phase, and inserted by the post processing phase into the DOM root of the document which has been encrypted using these key class objects. (See reference numbers 461, 462 of Fig. 4C.) --

Please replace the paragraph that begins on Page 35, line 14 and carries over to Page 36, line 15 with the following marked-up replacement paragraph:

-- A key object 535, 536, ... 539 will exist in a particular key class object 530 for each community member within the key class 531. Recall that a key object 500 or 510 is created for

each DN 501, and that each such key object includes an encrypted symmetric key 503. Thus, a key class object 530 for a key class 531 having 3 community members will include 3 key objects 535, 536, 539, and therefore will have 3 different encrypted symmetric key values 503 (that is, a different symmetric key value for each community member). For the employee record example where the individual employee, managers, and HR department employees comprise the 3 members of the authorized community for viewing current salary information, key class object 530 will include key objects with distinct encrypted keys 503 for each of these members. These 3 different symmetric key values are created from the single unencrypted key value 523 stored in the preprocessing key object 520. The public key 505 from the key object for each community member is used to generate the different symmetric key values. To decrypt the curr_salary information, the processing on behalf of a member of the managers group locates the managers key object among objects 535, 536, 539 by comparing the managers group DN to DN values 501, retrieves the encrypted symmetric key value 503 from the appropriate key object, and decrypts this symmetric key using the private key for the managers group. This decrypted key can then be used to decrypt the curr_salary information. Similarly, when a member of the HR department wishes to access the curr_salary, the DN for the HR group is compared to the DN values in objects 535, 536, 539 to locate the key object for the HR group. The encrypted key value 503 is then retrieved from that key object, and decrypted with the HR group's private key. This decrypted symmetric ~~private~~ key is then used by the HR group member to decrypt the curr_salary value. --

Please replace the paragraph that begins on Page 48, line 12 and carries over to Page 49, line 14 with the following marked-up replacement paragraph:

-- The processing of Fig. 7B begins upon completion of the processing of Fig. 7A. This processing may occur as part of the augmented XSL processor of the present invention, or may be performed by a transcoding proxy of the prior art (see the description of a transcoding proxy 615 in the discussion of Fig. 6, above). Block 752 applies a style sheet rule to the source document 400. When the pattern of a style sheet rule matches an element of the source document, Block 754 asks whether this template calls the existing XSL value-of method. If not, processing of the rule continues according to the prior art, and control transfers to Block 759. According to the present invention, the value-of method is preferably invoked for each element in the source document, in order to apply selective encryption to each element as needed. This may be accomplished by applying a style sheet that copies all input values to an output document being generated. When the value-of method is invoked by the template rule, Block 756 retrieves (i.e. obtains a pointer or reference to) the previously-instantiated data policy object for the data element which has matched the template rule. The overriding value-of method of this data policy object is executed at Block 758, performing any appropriate transformations that have been coded within this method. According to the present invention, the code of the overridden value-of method is written to determine whether the policy object specifies that the data element is to be encrypted (as described above with reference to Block 715), and if so, to insert encryption markup tags around the element. The encryption markup tags preferably use a syntax such ~~as "<encrypt:data class = "n"">~~ and as "<encrypt:data class = "n">" and "</encrypt:data>" (as illustrated in Fig. 4B at 422 and 424), where the value of "n" is set to the identifier of the key

class object that was associated with this policy object in Block 745 of Fig. 7A. This process of applying style sheet rules to mark up the source document 400 is repeated by iterating Blocks 752 through 759 until the source document has been completely processed (i.e. until the test at Block 759 has a positive result), after which the processing of Fig. 7B ends. --

Please replace the paragraph that begins on Page 49, line 15 and carries over to Page 50, line 5 with the following marked-up replacement paragraph:

-- Fig. 4B illustrates an example of the result of completing the processing of Fig. 7B upon source document 400. Note that the content 420 of Fig. 4B represents interim information which is created and used internally. The information is never exposed in this form. (See Fig. 4C for an illustration of the information that is exposed externally.) Markup tags 422 and 424 have been inserted to bracket the security-sensitive values of the curr_salary and medical_condition document elements. A first key class is to be used for encryption of the curr_salary value, and a second key class is to be used for the medical_condition value, as indicated in the markup tags at 423 and 425, respectively. Fig. 4B further illustrates the organization of these key class objects. As shown at 430, key class "1" has an associated algorithm (shown in the figure as type="3DES") and key length (shown in the figure as len="168"), and includes key objects 431, 432, 433 for the three members of the associated community. Key class "2" is similar, using a different algorithm and key length (see 440), and specifying key objects 441, 442 for two community members. --

Please replace the paragraph on Page 50, lines 6 - 13 with the following marked-up replacement paragraph:

-- Note that the "tempkey" elements 434, 444 of Fig. 4B depict examples of the unencrypted symmetric key 523 that will be used to create a different encrypted symmetric ~~key 523~~ key 503 (shown in Fig. 4B and 4C as the values of the "Ekey" attribute) for each community member of the associated key class. Note also that the KeyIdentifier values and these Ekey values depicted in the key classes (in both Figs. 4B and Fig. 4C) are merely to allow visual representation. In an actual implementation, this information is preferably encoded as binary values using "base 64" rules as known in the art, such that the result contains only printable characters that are allowed in the context of XML attribute values. --

Please replace the paragraph on Page 53, lines 1 - 14 with the following marked-up replacement paragraph:

-- Block 780 encrypts the generated symmetric key 523 separately for each community member (that is, for each distinct DN within the community) authorized to view the associated document element. This is performed by accessing each key ~~object 510~~ object 500 (as stored in field 535, 536, ... 539 of key class object 530) defined for the current preprocessing key class, and for each key object, (1) retrieving the public key 505 from the X.509 certificate 502a, (2) using this public key 505 to encrypt the symmetric key 523 using the encryption algorithm and key length stored at 532 and 533, respectively, and (3) storing the resulting encrypted key in field 503 of the key object. This will result in one encrypted copy of the symmetric key per community member having a separate DN 501 and X.509 certificate 502a. (In other words, when a community member is a group representing multiple individuals, then one encrypted copy of the plaintext symmetric key 523 is generated for the entire group and is associated with the group's

DN.) To save space, the preferred embodiments then replace the X.509 certificate 502a with its corresponding KeyIdentifier 502b (such that format 500 is replaced with format 510), which in combination with the distinguished name 501 allows identification of the specific certificate which was used during encryption. --

Please replace the paragraph on Page 53, lines 15 - 17 with the following marked-up replacement paragraph:

-- Block 785 then inserts the key class object 530 into the root of the DOM, as illustrated by the presence of key class objects 461, 462 ~~in the what~~ in what may be considered the root area 460 of the output document 450 of Fig. 4C. --

Please replace the paragraph on Page 54, lines 16 - 20 with the following marked-up replacement paragraph:

-- While the selectively-encrypted document example shown in ~~Fig. 4B depicts~~ Fig. 4C depicts the element tags as having been left unencrypted, it may happen in a particular situation that it is desirable to encrypt the tags themselves as well as the data value(s) enclosed by the tags. To accommodate this possibility, the associated policy object may be written such that it places the encryption tags (see 422, 424 of Fig. 4B) surrounding the element tags rather than surrounding the element value. --

Please replace the paragraph that begins on Page 60, line 16 and carries over to Page 61, line 7 with the following marked-up replacement paragraph:

-- Another preferred embodiment is defined for the situation where either (1) a user's workstation has insufficient processing power to perform the decryption process of the present invention, or (2) it is desired to avoid program code changes on the client workstation. Thus, a client proxy performs the decryption process on behalf of the user (or on behalf of an application executing on the user's workstation). The logic with which this preferred embodiment may be ~~implemented in depicted~~ implemented is depicted in Figs. 9A and 9B. The user in this scenario uses a standard Web browser client application (such as browser client 675 of Fig. 6) executing on his workstation or a standard program client (such as program client 680), and requests a selectively-encrypted XML document from a proxy (such as client proxy 655) acting on behalf of the browser client. Through use of this client proxy 655, this preferred embodiment enables serving selectively- encrypted document content to browser clients 675 or program clients 680 without requiring any program changes or additional software operating on the client device. --

Please replace the paragraph on Page 67, lines 7 - 19 with the following marked-up replacement paragraph:

-- The process of decrypting an element on behalf of the client 675 begins at Block 948, where the proxy 665 expands the group membership of those DNs which represent groups in the key class of this element. Referring to the example document in Fig. 4C, the processing of Block 948 comprises locating the key class identifier 456 when processing encrypted element 452, then finding the DNs of groups 471, 473 from the key objects within the key class 461 having the ~~identifier 453 (see~~ identifier 456 (see 463) which is located on the encrypted element tag, and then determining the membership of these groups having common names "managers" 470 and "hr"

472. When an LDAP directory is used for storing group membership information, as has been described, determining the membership comprises issuing an LDAP command for each DN assigned to a group, where that command retrieves the DNs of the individual members of the group. (As will be obvious, when a data repository other than a directory is used for storing group membership information, the retrieval command appropriate to that repository is used.) Alternatively, a query can be performed in the form of "Is this (individual DN) a member of this (group DN)?". --

Please replace the paragraph on Page 78, lines 3 - 16 with the following marked-up replacement paragraph:

-- Returning now to Fig. 10A, Block 1024 resumes the user's processing, after the clerk processing of Fig. 10C has completed, by receiving the key objects (i.e. within key class objects) which were passed. Optionally, the certificate which was passed (or its key identifier) may also be returned so that the requester can easily locate the corresponding private key on its local key ring or chain (given that a requester may have more than one certificate and private key). If the returned information was digitally signed by the clerk, the user first verifies the digital signature to ensure that the information was not created by an impostor and has not been altered. (If the session between the user and clerk is a mutually-authenticated secure session, then the signing by the clerk, and verification by the user, is not required.) The user then decrypts the symmetric key from each key class object returned from a group clerk (Block 1026) using the user's private key. (Or, if the keys are returned unencrypted on a mutually-authenticated secure session, this decryption in Block 1026 is not required: these unencrypted keys will be used directly for

decrypting elements of the associated key classes.) The user will now be able to decrypt the elements for that key class, as ~~will be~~ will now be described with reference to Fig. 10B. --